FDHS C++ Programing Club

C++ 基礎入門教學講義

108 學年度 | 復旦程式設計班專用

班級：________________ 姓名：________________

復旦程設班

歷屆幹部世系表

職務 屆數	班長	副班長	總務長	教學長	進階 教學	初階 教學	初階 助教	進階 助教	國中部 負責人	班級 秘書
創始	鍾鎵駿		黃先鼎　陳耀恩　王宗仁　李修豪							
1	黃琪文	宋耘慶	李宛庭	羅子原						
2	童宇宏	張任廷	蔡語謙	洪冠群					蔡語謙	
3	陳柏昇	曹竣傑	徐瑞陽	林宏俊						陳柏宇
4	蔡語謙	方喬	王維澤		張鈞諺	陳皇佑				
5	張瀚文	高程昱	林承緯	何培魁	鄭豫澤	姜漢暘				
6	石宇辰	陳博威	李智霖		李驊諺	徐惇楠	楊景鈞 鍾孟辰			
7	古景睿	呂恆甲	林宥廷	邱秉辰	鍾孟辰	許愷洌		傅靖芸		
8 **現任**	吳振安	邱秉辰	陳睿岩		吳樹棋	許愷洌	陳允浩 楊登棋	施羿廷		

前言

首先，歡迎大家加入程設班！！

復旦程設班的前身，是訓練資訊學科能力競賽比賽同學的計畫。但學長們對於程式設計的渴望，在他們國中時，就陸續請求加入訓練，人數開始增加。後來鍾鎵駿、黃先鼎、王宗仁、陳耀恩、羅子原、黃琪文、張任廷、洪冠群、蔡語謙等學長姐，努力宣傳招生，設計許多的社團活動，讓程設班同學樂於互相交流，一同去參與比賽，於是程設班有了今日的規模。今日連教學講義、招生微電影都誕生了。這一切都不得不佩服這學長姐的努力和對程設班的熱愛。

學習程式設計是我們的主要目標。因為你想、你喜歡寫程式，今日才有你的加入。但更希望你能在此，感染到學長們的熱情、進而喜歡來到這個社團。在你要離開這個班時，期待你能帶著滿滿的知識和回憶，自豪的說：「我很開心來到這裡，復旦程設班，讚啦！」

陳宏智 老師 104/08/31

學長的話

給學弟妹們：

轉眼間，程設班從創立到現今已有一段時間了，每一屆都有為這個班付出的神話存在，從班規到現今有規模的教學制度，從無到有的生出招生影片，讓我這個老人家感到很欣慰，更開心的是，歡迎大家有緣能加入這個程設班的大家庭，這個班～看似普通平常，只是寫寫程式而已；但～這就錯了，這個班能給大家的東西是極其稀有的，包括學長姐們的無私分享自己的求學歷程，相信這在你們自己班上是很難遇到的。總而言之，加入復旦程設班是絕對不會讓你後悔的，這是個有緣讓大家相聚的一個復旦私下特色社團，從第一屆的學長姐們篳路藍縷的創班艱辛，到現在制度規模完全的一個程設班。希望大家能在這個班上學到很多在學校學不到的東西，加油！！

創班班長 鍾鎵駿 104/08/31

編者的話

歡迎大家加入程設班的奇幻旅程

首先我必須要說一句，復旦的環境實在是太小了，但我們不能被尛復旦限制，記住：要站得更高才能看得更遠，在這裡，程設班就是你開闊視野的第一步，這一年內我們會學習如何 coding，也會參加相關比賽，希望大家能從中學習，祝大家玩得愉快，GL HF!。

2nd 教學長 洪冠群 104/08/31

歡迎加入無限 coding 的世界~

首先，寫程式”不等於”宅男，更不等於”工具人”，這完全是習慣、個性，與女生正不正的問題。一位程式的愛好者，是喜歡多方探索的，生活中的眉眉角角都能激發我們的思考，我們運用程設班的所學，將腦中的想法轉換為文字、將心中的創意實現於程式，於是我們一腳踏進了無限 coding 的世界！！各位學弟妹們，歡迎加入程設班，希望你們在未來能透過老師、學長姊，以及各種比賽的歷練，親手打造出描繪創意的夢想之筆。

2nd 副班長 張任廷 104/08/31

修訂序

再次歡迎大家加入復旦程設班的大家庭

「持之以恆，不輕言放棄」是我最想送給程設班各位學員們的一句話。從國一下開始跟你們一樣一無所知的接觸 C++，在學習的路上，我也遇到很多瓶頸、挫折，雖然這些絆腳石令人沮喪、難以前進，我還是堅持自己的信念不斷地去練習、克服，最終得到了豐碩的成果，給了我繼續向前行的力量。大家一定要記得，程式是一門學不完的科目，它就像是一個世界一樣，等著你逐漸去探索、發掘，而初階班的課程就像是打開程式世界大門的鑰匙，重要性不容小覷。

在這個版本的修訂中，我們改善了版面的編排，增加了不少生活化的題目，使得內容更加簡潔明瞭也增添了不少趣味性。感謝編者及之前的修訂者們，也希望各位學員在這一年能多多包涵，我們一起努力，讓程設班變得更好!

7th 教學長 邱秉辰 中華民國 107 年 6 月

再修訂序

希望大家不要忘了來程設班的初衷

如同創班班長常講的話，不論你是來程設班學程式、交友、當電神、享受宅男的氣氛(?)，不管你的目的是什麼，都希望各位在程設班的一年，能夠帶走一些東西，不要在程設班虛度光陰，因為各位都是在淘汰別人的基礎上進來程設班的。如果你星期五都浪費四小時在這裡而什麼都沒學到，會很對不起那些想學卻沒考上的人。

同時，也希望各位回家可以多多練習線上的 Online Judge 或是我們出的作業。寫程式就像寫數學、物理、化學(no offense)一樣，概念可能大家都懂，不過少了實際操作的演練和 debug 的過程，等到真正用到時，可能連開頭都無法。

最後，希望大家畢業多年後，回想當初程設班的時光，都是充實且豐富的！

8th 初階教學 許愷洌 中華民國 108 年 8 月

幹部的話

程設班的創立初心是希望想要寫程式的人都能有一個地方寫程式

如今程設班的規模從最初的不到五人到今天五,六十人，但是那份初心不變。而我們也希望每個人在程設班除了寫程式外，能把程設班看作是一個家，因此我們沒有學長學姐制。大家除了自己寫程式之餘也能與別人多多交流，不管是上課或玩晚餐遊戲還是聖誕活動等等大家都能認真參與，珍惜在程設班的每分每秒。最重要的是希望大家能在這個班開心的學寫程式，因為不管程式寫得好不好，至少寫程式是開心的是好玩的，期望大家都能有這樣的感覺。

6th 助教 7th 進階教學 唯二女幹部 鍾孟辰

中華民國 107 年 7 月 25 日

目次

Chapter 1 ----- Welcome to C++

Chapter 2 ----- Variables and Data Types

Chapter 3 ----- Operators and Conditional Statements

Chapter 4 ----- Loops

Chapter 5 ----- Array and String

Chapter 6 ----- Functions

第五版

5TH EDITION

為了輔助程式設計班教學之用，依照各屆學員的盲點及常問的問題，以重點歸納的方式編排此講義，內容中為求淺顯易懂，用字遣詞稍有偏差，待學員往後自行釐清觀念。

Copyright © 2019 Fudan C++ Programming Club.
All rights reserved

Chapter 1.

Welcome to C++

1-1 程式的基本架構

下列是一個極簡單的程式碼，本節將簡單介紹此程式碼內的包含的東西及意義

code 1-1.cpp

```
//code 1-1
#include<iostream> // 含括<iostream>函數庫
using namespace std; // 不加很麻煩
int main() // 主程式碼
{
    cout<<"Hello, World!"<<endl; // 輸出
    system("pause"); // 顧名思義暫停
    return 0; // 回傳值
}
```

#include<iostream> // 含括 iostream 函數庫

引入的函數庫，類似字典一般，讓每行指令知道它的功能

using namespace std; // 使用名稱空間：std

不想用的話需在每一次輸入、輸出等前加上 std:: (p.s.真的很麻煩 真的)

int main( ) { } // 程式主體

主要程式碼的撰寫處

活動 1-1

請在自己的電腦上撰寫如上的程式碼，且條件如下表格並編譯成四個不同的執行檔，

請執行.exe 並將觀察結果填入下表格。

system(“pause”); / return 0;	有 system(“pause”);	無 system(“pause”);
有 return 0; 作結		
無 return 0; 作結		

※注意請至單獨點擊.exe 檔執行，在編譯器使用 F11 效果不顯著

從活動 1-1 可得出一個結論，程式.exe 執行完會**自行關閉**，需要**使用 system(“pause”)暫停**，但若只有純粹解題需求，不需要用到.exe 的話，可不必添加。由於目前使用的編譯器為 Dev C++，主程式 **return 0;**的有無是沒有太大的區別的，但是在其他編譯器上，**return 0;**作結是必要的，為了避免不必要的麻煩，請在程式結尾加上。

反思

如果把#include<iostream>或 using namespace std;刪除的話將會得到什麼錯誤訊息呢？而這些訊息又是什麼意思？

本節重點及易錯細節

1. 在程式碼中「空白(space)」及「換行(enter)」對電腦而言是**沒有意義**的，因此一個程式設計者要**善用空白及換行整理程式碼**，**簡潔明瞭的程式碼**將是未來 debug 的關鍵。
2. 而在程式碼中真正做為一項指令結束的是**分號(;)**，因此在每個命令下完記得要以**分號作結**。
3. 忘記打分號: 程式會顯示 [Error] expected ';' before 'XXX' ，簡單說，你少什麼，他就說他期望在 XXX 前有什麼
4. #include<iostream>、using namespace std; 超級容易拼錯

1-2 基本的輸出

在程式中若要與使用者作互動，最直接的方式便是**輸入**與**輸出**的運用，不過目前還沒教到輸入，僅在這裡概述輸出。

如果要輸出一段文字，我們使用 **cout 和 <<**

例： cout<<”一段文字” <<endl;

```
一段文字

--------------------------------
Process exited after 0.05192 seconds with return value 0
請按任意鍵繼續 . . .
```

endl 是 end of line 的簡寫，是換行的意思，我們可以看到在"**一段文字**"下面有一行是空白的，那就是 **endl**。

課堂練習：

請撰寫一程式，請輸出自己的名字。

範例輸出：

```
超級無敵帥

--------------------------------
Process exited after 0.01958 seconds with return value 0
請按任意鍵繼續 . . .
```

Chapter 2.

Variables and Data Types

2-1 變數的命名

命名一個變數，需要注意下列事項：

1. 使用**識別字**
2. 字母的大小寫是不同的且須半形字
3. 數字不得為開頭且符號只能用__(底線)
4. 中間不得有空格

※ 識別字與關鍵字的差別

識別字：根據程式需求自行定義的名稱

關鍵字：程式語言保留的關鍵字，作為指令與特殊用途

定義識別字時不可與關鍵字相符，在 Dev C++中，關鍵字通常會以粗體呈現

例：

```
or and xor switch if else
misca musca John Cena
```

2-2 變數的宣告

1. 在日常生活中，我們會運用如 x,y 之類的代數帶入數字以簡化數字的運算，在程式中也有諸如此類的概念
2. 程式語言中，我們將數字、字元等資料存放在一個代替的名稱中，使程式設計師自由的運用，因此我們透過**宣告變數**的動作來**讓每個名稱有意義**
3. 未宣告變數前電腦是不認識他的，所以變數在使用前都必須要先宣告

```
資料型態 變數名稱;
資料型態 變數名稱=初始化值;
```

4. 在宣告名稱前，我們會加一個存放的資料型態(下例為 int、char)來定義他，電腦中的每筆資料都由數字構成，而資料型態就是這個數字的範圍，若超過這個範圍就會得到亂碼，稱為溢位(overflow)。
5. 在定義資料型態為字元的變數時，須以雙單引號夾字元作定義('a')。

```
int Fudan;
char I_love_Taiming='?';
```

6. 可用”,”(逗號)做連續宣告(eg：若要宣告整數變數 a 及 b，可將 int a；int b；合併為 int a,b；)

重要觀念釐清

int b=5 的涵義是將 5(右)設定給 b(左)，不同於數學常用的等於，此處一個等於是“將...定義為...”的意思。 而在數學裡的”=” ，在程式裡我們用”==”來表示，同學需特別注意。

2-3 資料型態的表列

資料型態	**位元組**	**表示範圍**
int	4	-2147483648 到 2147483647
unsigned int	4	0 到 4294967295
long long int	8	-9223372036854775808 到 9223372036854775807
unsigned long long int	8	0 到 18446744073709551615
short int	2	-32768 到 32767
unsigned short	2	0 到 65535
float	4	1.2e-38 到 3.4e38(1.2*10^-38 到 3.4*10^38)
double	8	2.2e-308 到 1.8e308(2.2*10^-308 到 1.8*10^308)
char	1	-128 到 127(256 個字元)
bool	1	0 或 1

※ double 雖然可以存很大的數，但是其精確度大概只有 15 位左右

2-4 基本的輸入

方法 使用 cin 與>>的搭配

例 cin>>a; //將輸入的東西儲存進變數 a 裡面

若要連續輸入的話，可以用多個>>來完成

例 cin>>a>>b; //連續輸入變數 a 跟 b

※易錯重點：

輸入多變數時不需打空格，例：~~cin>>a>>” ”>>b;~~

或是在結尾作換行，例：~~cin>>a>>b>>end;~~

課堂練習：

請撰寫一程式，輸入兩變數為班級、座號

輸出為 XXX 班 XX 號

範例輸出：

2-5 輸入與輸出的範例程式碼

```
//code 2-5
#include<iostream>
using namespace std;
int main()
{
    unsigned int year;
    cin>>year;
    cout<<"今年是U賀復旦第"<<year<<"周年"<<endl;
    return 0;
}
```

輸入:2019 01 01

輸出:今日為 **2019** 年 **1** 月 **1** 日

1. DOS 視窗(一個黑黑的視窗)，是**使用者與電腦互動的介面**。在這個介面中，**游標(閃爍的白色底線)**，將決定**輸入**及**輸出**的位置。因此在使用 C++進行應用時，程式設計師應注意游標的位置。

2.如何改變游標出現的位置，或做相關應用，就待各位自行上網發掘了。

課堂練習:

請撰寫一程式，輸入西元年分，請輸出民國年分

範例輸出：

```
1911
民國1年

--------------------------------
Process exited after 4.141 seconds with return value 0
請按任意鍵繼續 . . .
```

2-6 型態錯誤及溢位範例

```
//code 2-6
#include<iostream>
using namespace std;
int main()
{
    int i,i1;
    double d,d1;
    int a;
    cin>>i>>i1;
    cout<<"int型態只能顯示整數位"<<i/i1<<endl;
    cin>>d>>d1;
    cout<<"double型態能顯示至小數位"<<d/d1<<endl;
    cin>>a; //請輸入大於2147483647或是小於-2147483648的整數
    cout<<a<<endl;
    return 0;
}
```

輸入：	輸出：
100 3	int 型態只能顯示整數位 33
100 3	double 型態能顯示至小數位 33.3333
7777777777	2147483647

2-7 跳脫字元的認識

在單引號中，有些特定的字元搭配反斜線(\)就會有特殊的功能(指令)，而這便稱為跳脫字元，詳見右方表格。

- 輸出雙引號範例：

```
//code 2-7
#include<iostream>
using namespace std;
int main()
{
    cout<<"\""\n";
    return 0;
}
```

跳脫序列	意義
\a	警告音
\b	倒退一格
\n	換行
\r	歸位
\t	跳格
\0	字串結束字元
\\	反斜線
\’	單引號
\”	雙引號

Chapter 3.

Operators and Conditional Statements

3-1 常見的運算子

1. 設定運算子 =

Def. 將左方**設定**為右方數值

例：a=b; //將 b 的值設定給 a(並非 a 等於 b)

2. 括號運算子 ()

Def. 決定運算的先後順序

例：a-(b+c); //(b+c)要先算

3. 算數運算子

+	-	*	/	%
加法	減法	乘法	除法	取除以右邊的餘數

4. 關係運算子

>	<	>=	<=	==	!=
大於	小於	大於等於	小於等於	等於	不等於

5. 邏輯運算子

<table>
<tr><th>&&</th><th>||</th></tr>
<tr><td>and 且 (都要成立)</td><td>or 或 (其中一個成立)</td></tr>
</table>

6. 一元運算子

+	-	!	~
正數	負數	not 否	取 1 的補數

優先順序：!(not) > &&(and) > ||(or)

7. 三元條件運算子

(條件) ? A : B

如果條件為真，則 A

反之，則 B

8. 運算子的結合

	a=a+b	a=a-b	a=a*b	a=a/b	a=a%b
可簡化為	a+=b	a-=b	a*=b	a/=b	a%=b
定義	a+b 的值給 a 存取	a-b 的值給 a 存取	a*b 的值給 a 存取	a/b 的值給 a 存取	a%b 的值給 a 存取

	a 遞增	a 遞減
可寫為	a++	a--

3-2 判斷

if 判斷

1. if：

※當小括號內的判斷條件成立後

就會執行大括號內的程式

※只能有一個 if

```
if(判斷條件)
{
    程式碼
}
```

2. else if：

※可有多個 else if

※前面需先有 if 假設，當 if 假設不成立

時才會執行 else if 判斷

```
if(判斷條件)
……
else if(判斷條件)
{
    程式碼
}
else if(判斷條件)
{
    程式碼
}
……
```

3. else：

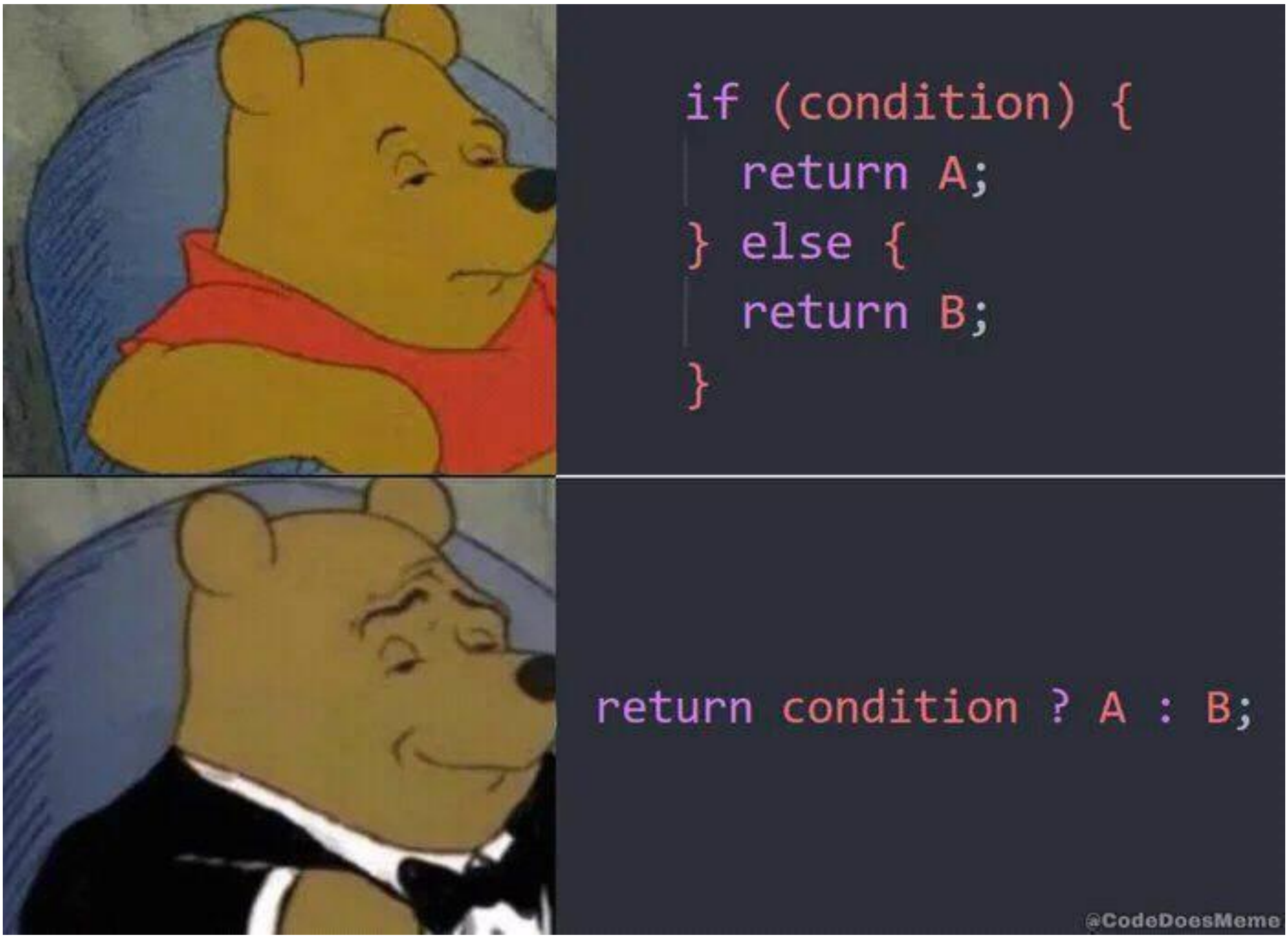
※當上方 if 和 else if 皆不成立後才會執行

※只能有一個 else

※else 不需判斷

```
if(判斷條件)
……
else if(判斷條件)
……
else
{
    程式碼
}
```

if 和 三元條件運算子的差別



圖：https://www.facebook.com/ProgrammersCreateLife/photos/a.241809332534619/2470026513046212/?type=3&theater

3-3 程式碼範例

```
//code 3-4 關係、邏輯運算子範例
#include<iostream>
using namespace std;
int main()
{
    int a,b;
    cin>>a>>b;
    if(a+b>=120 && a>=60 && b>=60)
        cout<<"PASS"<<endl;
    else
        cout<<"FAIL"<<endl;
    return 0;
}
```

輸入：
80
80

輸出：
PASS

動動腦：
如果判斷式用數學常見的
例：0<=a<=7 能不能夠成立?

```
//code 3-4 一元運算子範例
#include<iostream>
using namespace std;
int main()
{
    short s=5;
    bool b=true;
    cout<<"s="<<s<<endl;
    cout<<"b="<<b<<endl;
    cout<<"~s="<<~s<<endl;
    cout<<"!b="<<!b<<endl;
    return 0;
}
```

輸出：
s=5
b=1
~s=-6
!b=0

動動腦：

true 代表什麼?

若把它印出會得到什麼?

false 又代表什麼?

若把它印出會得到什麼?

小遊戲------運算子的結合：

```
//code 3-4 小遊戲運算子的結合
#include<iostream>
using namespace std;
int main()
{
    int a,b,ans,ans2;
    cout<<"猜猜a+b??"<<endl;
    cin>>a>>b;
    cout<<"a="<<a<<" ,b="<<b<<endl;

    a+=b;
    b+=a;
    a+=b;
    b+=a;

    cout<<"after a+=b b+=a a+=b b+=a"<<endl;
    cout<<"Guess the answer (a+b)?"<<endl;

    ans=a+b;
    cin>>ans2;
    if(ans==ans2)
        cout<<"AC"<<endl;
    else
        cout<<"WA"<<endl;
    return 0;
}
```

你猜對了嗎？

Chapter 4.

Loops

4-1 迴圈的介紹

1. 假設今天要計算 1+2+3+4+5+6+7+8+9，可以直接鍵入數字到程式內，但若今天要計算 1+2+3+...+1000 呢?這時候我們就要**運用迴圈**幫助我們更有效率地做大量的運算

2. 迴圈是重複執行某些敘述，直到滿足特定條件為止，又稱為重複結構

3. 迴圈主要分為三種

   - 用於固定次數:for
   - 用於不固定次數:while、do-while

(概念相去不遠，但使用上有些區別)

4-2 while 迴圈

```
while(條件)
{
    ……
}
```

1. 當括號內的條件成立，就進入迴圈(大括號)。
2. 括號內的條件可以是一個指令(eg: cin>>a)，也可是一個判斷(eg: 1>0)
3. 當走到迴圈結束的地方就會回 while 重新判斷，直到條件不成立才跳出。

範例程式碼：

4-2.cpp

```
//code 4-2
#include<iostream>
using namespace std;
int main()
{
    int a,b,c;
    while(cout<<"Please enter 3 integers: " && cin>>a>>b>>c)
    {
        cout<<a+b+c<<endl;
    }
}
```

4-3 do while 迴圈

```
do
{
    ……
} while(條件);
```

1.概念與 while 類似，但是是先執行敘述再判斷，所以不論如何**至少會執行一次敘述**

2.此處的 while 後面需有分號

3.常用於密碼輸入

範例程式碼：

```cpp
//code 4-3
#include<iostream>
using namespace std;
int main()
{
    int num,password=711172;
    bool flag=true;
    do
    {
        cout<<"Please enter your password: ";
        cin>>num;
        if(num==password)
            flag=true;
        else
        {
            flag=false;
            cout<<"Wrong password\n";
        }
    }while(!flag);
    cout<<"Right password\n";
}
```

while 和 do while 的差異

圖：https://www.facebook.com/ProgrammersCreateLife/photos/a.241809332534619/2490945784287618/?type=3&theater

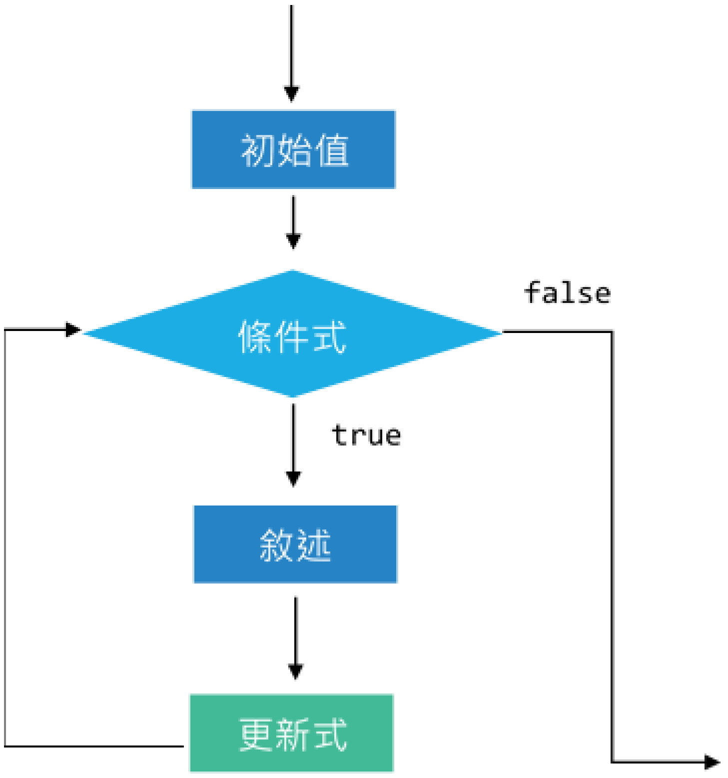
4-4 for 迴圈

1.又稱為計數迴圈(counting loop) ，是一個可以重複執行指定次數的重複結構

2. for 迴圈的標準使用格式

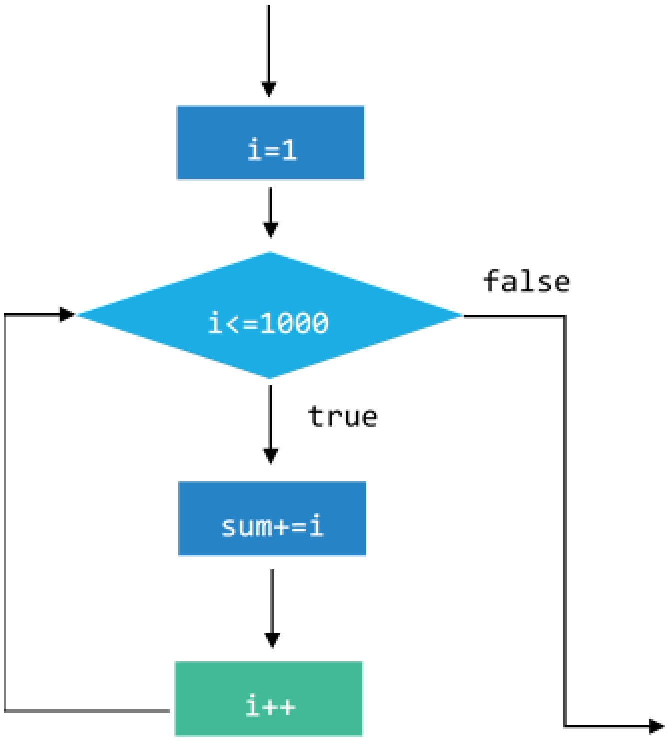
```
for(初始值;條件式;更新值)
    敘述;
```

```
for(初始值;條件式;更新值)
{
    敘述 1;
    敘述 2;
    ……
}
```



3. 範例(計算$\sum_{i=1}^{1000} i$)

```
//code 4-4
#include<iostream>
using namespace std;
int main()
{
    int i,sum=0;
    for(i=1;i<=1000;i++)
        sum+=i;
    cout<<sum<<endl;
}
```



分解步驟

i=1	sum=1
i=2	sum=3
i=3	sum=6
i=4	sum=10
i=5	sum=15
i=6	sum=21
i=7	sum=28
i=8	sum=36
i=9	sum=45
i=10	sum=55
i=11	sum=66
i=12	sum=78
i=13	sum=91
i=14	sum=105

………

輸出結果

500500

在進入 for 迴圈以前，整數變數 i 未設定，sum 為 0 在迴圈一開始將 i 設定為 1

程式碼 sum=sum+i 讓 sum=0+1=1，因此 sum 的值在執行完一次後變為 1 接著因為 i 並未超過 1000，因此迴圈繼續執行。

此時由於 i++的緣故，i 從 1 變為 2，sum 的值累加為 1+2=3，接著 i 變為 3，sum 再繼續累加為 6……，直到 i=1000 被 sum 累加完才停止。

4. 多重 for 迴圈

簡單來說，就是迴圈包迴圈，使用時應注意 **TLE(逾時)**的狀況

九九乘法表範例：

```
//code 九九乘法表
#include<iostream>
using namespace std;
int main()
{
	int col,row;
	for(col=1;col<=9;col++)
	{
		for(row=1;row<=9;row++)
		{
			cout<<col<<"x"<<row<<"="<<col*row<<"\t";
		}
		cout<<endl;
	}
}
```

1x1=1	1x2=2	1x3=3	1x4=4	1x5=5	1x6=6	1x7=7	1x8=8	1x9=9
2x1=2	2x2=4	2x3=6	2x4=8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
3x1=3	3x2=6	3x3=9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
4x1=4	4x2=8	4x3=12	4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
5x1=5	5x2=10	5x3=15	5x4=20	5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36	6x7=42	6x8=48	6x9=54
7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49	7x8=56	7x9=63
8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64	8x9=72
9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81

4-5 迴圈的中止

break;	continue;
直接跳離迴圈或 switch 結構	跳過 continue 指令之後的所有敘述，跳回迴圈的開頭繼續執行下個迴圈

※注意：別照字面翻譯 continue

範例：

```
//code 4-5
#include<iostream>
using namespace std;
int main()
{
    int a;
    while(cin>>a)
    {
        if(a!=0)
        {
            continue;
        }
        else
        {
            break;
        }
        cout<<"不會執行的程式碼\n";
    }
    cout<<"a=0"<<endl;
    return 0;
}
```

利用 break 及 continue

可以使迴圈的運用更加靈活

當輸入為 0 的時候才會結束。

若輸入不為 0，就會因為 continue

而回到 while 迴圈一開始

並且繼續輸入 a。

直到輸入為 0 時才會因為 break

跳出迴圈，但不論輸入是否為 0

都不會執行到迴圈內的 cout。

※利用 break 及 continue 配合 if 可以使迴圈的運用更加靈活

Chapter 5.

Array and String

5-1 一維陣列

之前我們使用的變數都只能一次儲存一個資料，但如果現在有 100 個資料呢？難不成要宣告 100 個變數嗎？這時候就需要用到陣列來幫助我們一次宣告大量變數。

陣列由很多相同資料型態的變數所組成，有點像是數列一樣連在一起，每一格都有自己的編號。

宣告格式

```
資料型態 陣列名稱[格數];
e.g. int num[8];
```

宣告時請注意：

1. 陣列的編號是**從 0 開始定義**
2. 除了宣告時的[ ]內代表格數外，往後使用時[ ]內的數字則代表編號

num[8]								
	num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]

陣列可在宣告時就指定每格的內容，若不指定則是殘值(亂碼)

e.g. `int num[8]={0,1,2,3,4,5,6,7};`

num[8]	0	1	2	3	4	5	6	7
	num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]

陣列是「單獨」作業的，無論 cin 或 cout 都是以單格來處理

e.g. `cin>>num[5];` (鍵入 9)

num[8]	0	1	2	3	4	**9**	6	7
	num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]

e.g. cout<<num[5]; //輸出為 9 被覆蓋過後，此時就並非是 5

常用範例：

Step 1
- 宣告了一個有n個空格的陣列a

Step 2
- 利用for迴圈設定陣列裡的值

Step 3
- 利用for迴圈輸出陣列

```cpp
//code 5-1
#include<iostream>
using namespace std;
int main()
{
    int n;
    while(cin>>n)
    {
        int a[n];
        for(int i=0;i<n;i++)
            cin>>a[i];
        for(int i=0;i<n;i++)
            cout<<a[i]<<" ";
        cout<<endl;
    }
}
```

輸入
3
1 2 3

輸出
1 2 3

反思：
如果直接輸出 a
請問出現的值代表什麼？

5-2 排序

Bubble sort：

Insert sort：

5-3 二維陣列

多個相同資料型態的變數，組成一維陣列; 同理，多個相同型態與大小的一維陣列可以組成二維陣列。

宣告：

```
先橫列(column) 再直行(row)
資料型態 陣列名稱[列數][行數]
e.g. int num[3][5];
```

		第 0 行 num[][0]	第 1 行 num[][1]	第 2 行 num[][2]	第 3 行 num[][3]	第 4 行 num[][4]
第 0 列	num[0][]					
第 1 列	num[1][]					
第 2 列	num[2][]					

```
陣列可在宣告時就指定每格的內容，若不指定則是殘值(亂碼)
e.g.  int num[3][5]={{0 , 1, 2, 3, 4},
                    {5 , 6, 7, 8, 9},
                    {10,11,12,13,14}};
```

		第 0 行 num[][0]	第 1 行 num[][1]	第 2 行 num[][2]	第 3 行 num[][3]	第 4 行 num[][4]
第 0 列	num[0][]	0	1	2	3	4
第 1 列	num[1][]	5	6	7	8	9
第 2 列	num[2][]	10	11	12	13	14

陣列是"單獨"作業的，無論 cin 或 cout 都是以單格來處理

e.g. cin>>num[2][3]; (鍵入 9)

		第 0 行 num[][0]	第 1 行 num[][1]	第 2 行 num[][2]	第 3 行 num[][3]	第 4 行 num[][4]
第 0 列	num[0][]	0	1	2	3	4
第 1 列	num[1][]	5	6	7	8	9
第 2 列	num[2][]	10	11	12	**<u>9</u>**	14

e.g. cout<<num[0][3]; //輸出 3 此處的 num[2][3]也不再是 13，而是 9

常用範例：

```
//code 5-3
#include<iostream>
using namespace std;
int main()
{
    int col,row;
    while(cin>>col>>row)
    {
        int a[col][row];
        for(int i=0;i<col;i++)
            for(int j=0;j<row;j++)
                cin>>a[i][j];

        for(int i=0;i<col;i++)
        {
            for(int j=0;j<row;j++)
            {
                cout<<a[i][j]<<" ";
            }
            cout<<endl;
        }
    }
}
```

二維(或多維)和一維陣列的宣告方式基本上差不多，需要注意的只有行和列的順序，順序弄錯的話程式可是會出錯的喔

輸入：

3 2

1 2 3 4 5 6

輸出；

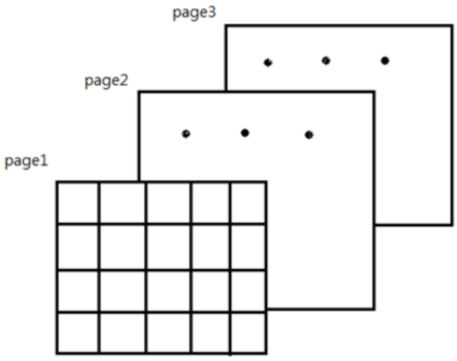
1 2

3 4

5 6

5-4 多維陣列

多維陣列的原理就跟一、二維一樣，以下是三維陣列簡單的圖示：



三維陣列的宣告也是一樣，如：int a[3][4][5];

當然你還可以宣告四維、五維……等多維陣列

但三維以上結構就較為複雜，四維以上更是需要憑著想像去撰寫

故在此僅提及並不多作介紹。

5-5 字元陣列及字串

1. 先前所教的字元變數只能存放一個字元，若要存取大量的文字、句子等，將會十分不方便

2. C 語言提供了字元陣列的處理方式，C++則提供了字串的格式，各有優缺點，可依自己喜好或情況選擇使用

3. C 型態的定義

字元常數 用單引號包覆 e.g. char c='i';

字串常數 用雙引號包覆 e.g. char c[ ]="I love fdcs"

4. 字串常數的記憶體配置，則會在結尾部分加上**結束字元'\0'**，同樣佔據一個記憶體位置，要注意(只要用雙引號包圍之字串，皆會加上\0)

I		l	o	v	e		f	d	c	s	\0

5. 字元陣列的定義及宣告

定義字串常數(字元陣列)所使用的方法，就如同先前陣列之定義方式

```
char 字元陣列名稱[長度];

char 字元陣列名稱[長度]= "字串常數" ;

char 字元陣列名稱[]= "字串常數" ;
```

6. 字元陣列的輸入

(1) 可以使用 cin 的方式來輸入字元陣列:

```
cin>>str;
```

請注意，如果輸入的範圍字元陣列範圍超過定義的範圍，將有可能發生預期外的錯誤。

(2) 也可以使用 cin.getline()，來輸入字元陣列:

```
char str[10];

cin.getline(str,10);
```

※sizeof( )的用法

```
//code 5-5
#include<iostream>
using namespace std;
int main()
{
    char str[]="hello c++";
    cout<<sizeof(str)<<endl;
    string s="hello c++";
    cout<<sizeof(s)<<endl;
    char str1[]="hello fdcs";
    cout<<sizeof(str1)<<endl;
    string s1="hello fdcs";
    cout<<sizeof(s1)<<endl;
}
```

輸出；

10

8

11

8

sizeof( )是將該陣列所佔的記憶體大小輸出，char 是每個字元皆佔一個位元組，而 string 則是固定為 8Bytes

5-6 C++型態的字串 String

C++的字串型態功能與 C 型態的字元陣列類似，但有一個不錯的優勢，就是他不需要先行定義字串的長度，就可以輸入了。

在使用 string 類別之前，請先引入標頭檔

```
#include<string>
```

1. 宣告格式

string 字串名稱;

string 字串名稱=” 字串內容” ;

```cpp
// code 5-6
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str0;
    str0="Hello C++";
    string str1="Hello FDCS";
    cout<<str0<<endl<<str1<<endl;
    cout<<str0.length()<<" "<<str1.length();
    cout<<endl;
}
```

輸出結果

Hello C++

Hello FDCS

9 10

程式碼重點

其中的 .length()，可以用來讀取 string 型態的長度，可以發現字串的長度皆是 9，並沒有像 c 字串一樣結束字元的存在。

比較：若是使用 sizeof()，則會輸出 string 所佔記憶體大小(8)

輸入格式

字串的輸入法同樣也**可以用 cin** 方式，但同樣，**遇到空格也會出現問題**，因此**可以使用 getline(cin,str);** 的方式**輸入有空格的句子**。

```
// code 5-6
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str0,str1;
    getline(cin,str0);
    cin>>str1;
    cout<<str0<<endl;
    cout<<str1<<endl;
}
```

輸入

Hello world

Hello world

輸出

Hello world

Hello

由這個例子我們可以看到使用 cin、getline(cin,string) **兩者之間的差別**

C++型態的特殊情況 cin.get();

```
// code 5-6
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int n;
    cin>>n;
    string s;
    getline(cin,s);
    cout<<n<<" "<<s<<endl;
}
```

我們想要在輸入字串之前先輸入一個數字，但卻在輸入數字之後直接執行了剩下的程式。

這個現象是因為，當我們輸入完 5(或任何整數型態)時，enter 鍵會附加一個\n 的字元，如同前面所述，會存入 buffer(緩衝區)，並在下次呼叫字元時存入程式中，此時字串 str 存的值變成\n，可以自行設計程式檢驗看看。

要解決這個問題，我們可以透過在 getline(cin,str);前面加上 cin.get( );這個函式的意義是吸入一個字元(注意，**是單一字元**)並且儲存( )中的變數，若為空白，則不會存入任何位址而吃掉。故改變後的程式碼如下:

```cpp
// code 5-6
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int n;
    cin>>n;
    string s;
    cin.get();
    getline(cin,s);
    cout<<n<<" "<<s<<endl;
}
```

輸入
5
Hello world

輸出
5 Hello world

cin.get( )這個函式不僅可以用在吸收多餘的字元，也可以拿來存取單一字元喔!

字串同樣也可以寫成陣列型態，以同時存放多組句子

例如: string str[3];

```cpp
// code 5-6
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str[3];
    for(int i=0;i<3;i++)
        getline(cin,str[i]);
    for(int i=0;i<3;i++)
        cout<<str[i]<<endl;
    cout<<str[2][5]<<endl;
}
```

輸入
I love C++
Hello world
FDHSCPP

輸出
I love C++
Hello world
FDCSCPP
P

Chapter 6.

Functions

6-1 函數是什麼

在介紹 C++的函數前，我們先來想一下「函數」到底是什麼？

數學上的方程式 y = 2x+1 可以用 y=f(x)=2x+1 來表示，這個 f(x)就是一個函數。我們分析一下這個 f(x)函數，可以發現它主要分成三個部分：引數 x、函數本體 y=2x+1、傳回值 y

而我們在 C++裡使用的函數，也包含了這三個要素，而且是缺一不可。

原型宣告的架構如下：

```
傳回值型態 函數名稱(引數型態) ; (函數可存放多個引數)
```

6-2 C++之中的函數

函數的部分有所謂的"void"型態，在 C++中，傳回值為 void 的函數，也就是沒有傳回值的意思；同理，如果引數為 void，那麼就代表這個函數不需要傳回值也可以運作。

在定義完了函數的原型後，我們可以開始寫函數裡面的內容，如以下範例是不是簡潔乾淨美觀大方賢慧呢?

函數通常用於簡化程式碼，各位可以試著把一些先前寫過的程式利用函數的形式簡化。另外某些特別的演算法需要搭配上函數的特性才能實現，雖然這本講義裡並不會教，但是那其實是程式設計的精華所在，有興趣的話可以去找些書或是學長問問看。

```
// code 6-2
#include<iostream>
using namespace std;
void Triangle(int);//宣告Triangle函數
int main()
{
    int n;
    while(cin>>n&&n>0)
        Triangle(n);
    return 0;
}
void Triangle(int n)//函數本體
{
    for(int h=n;h>0;h--)
    {
        for(int w=h-1;w>0;w--)
            cout<<" ";
        for(int w=2*(n-h)+1;w>0;w--)
            cout<<"*";
        cout<<endl;
    }
}
```

<-也可以不宣告函數

直接把下面的本體寫在上面。

6-3 有多個引數的函數

函數其實是可以允許有多個引數的，甚至是不同引數可以有不同的回傳值(有個專有名詞叫多載 overload)，把上一個程式碼稍微修改一下。

```
// code 6-3
#include<iostream>
using namespace std;
void Triangle(int,char);//用","隔開各個引數
int main()
{
    int n;
    while(cin>>n&&n>0)
    {
        Triangle(n,'*');//把三個符號分別傳入函數
        Triangle(n,'#');
        Triangle(n,'.');
    }
}
void Triangle(int n,char word)//n是層數，而word是要印出的字元
{
    for(int h=n;h>0;h--)
    {
        for(int w=h-1;w>0;w--)
            cout<<" ";
        for(int w=2*(n-h)+1;w>0;w--)
            cout<<word;
        cout<<endl;
    }
}
```